1. Write a program that subtracts two numbers so that the result is negative. Then perform another subtraction that gives zero in output and check the SF and ZF flags.

Code:

.model small

.stack 100h

.data

num1 db 0x05

num2 db 0x08

num3 db 0x06

num4 db 0x06

.code

main proc

mov ax, @data

mov ds, ax

;first subtraction

mov al, num1

sub al, num2

;second subtraction

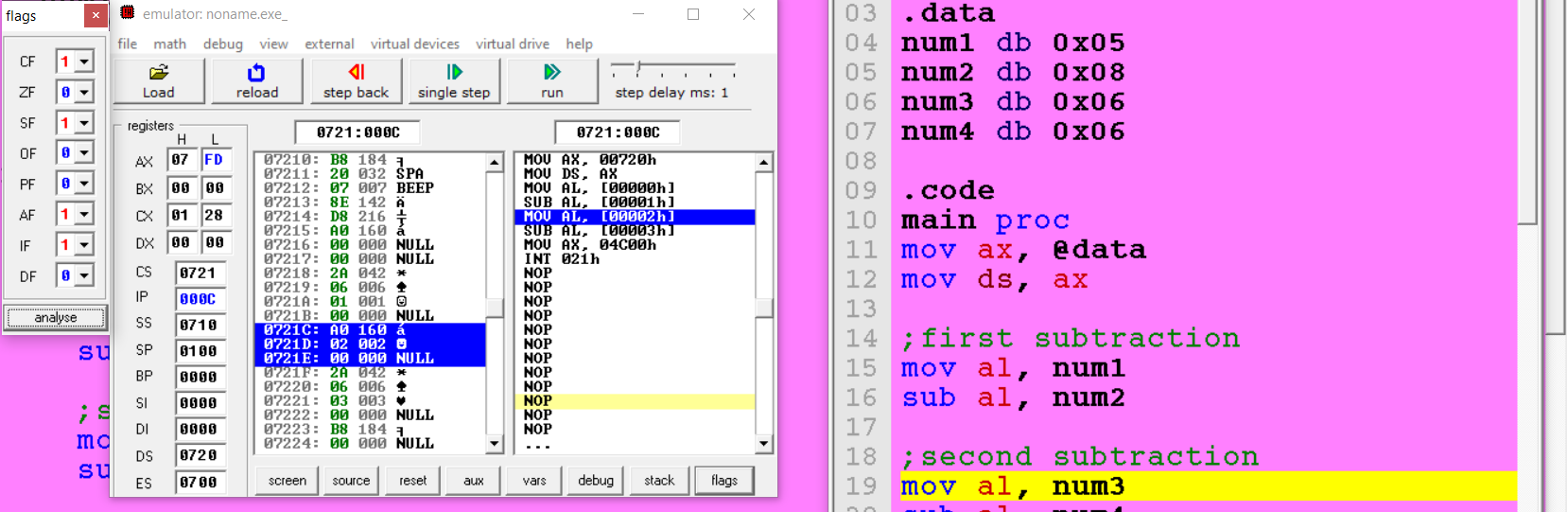
mov al, num3

sub al, num4

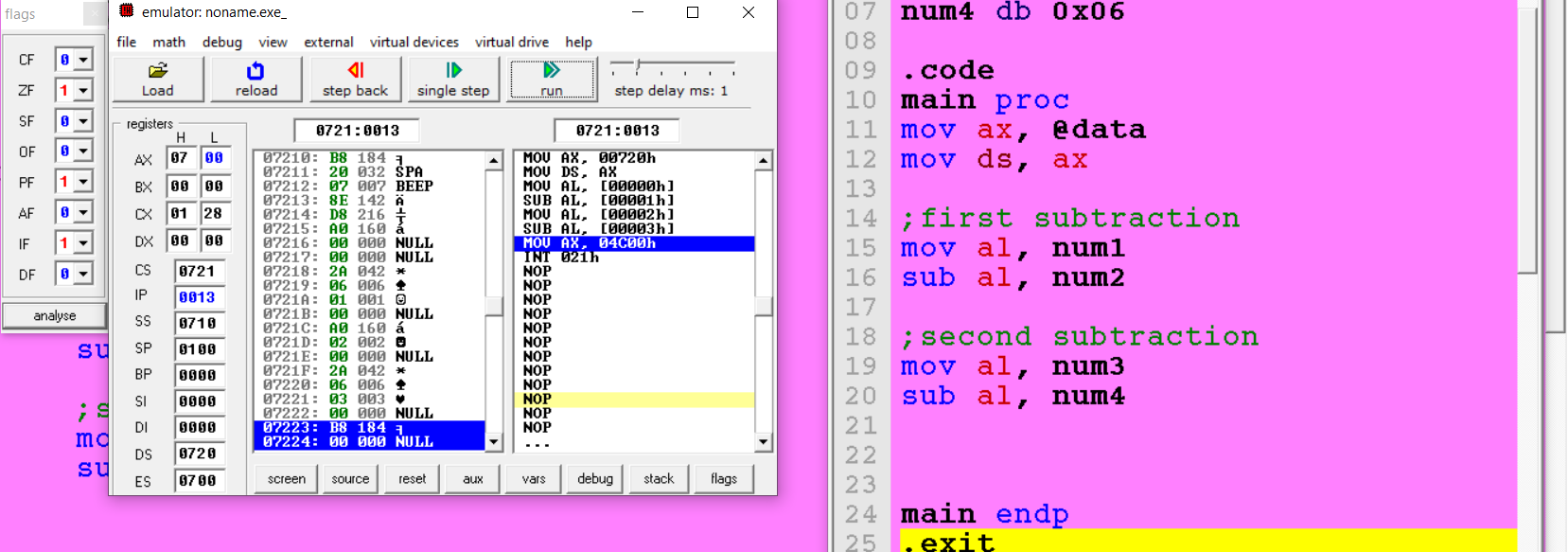
main endp

.exit

First subtraction:



Second Subtraction:



1. Write a program that stores the given two 32-bit numbers into the current data segment at offset: 0x1000 and 0x1008, respectively. The program then calculates the sum of these numbers and stores it at the offset: 0x1010. Numbers: 0x1F540398, 0xC0A1F02E.

Code:

.model small

.stack 100h

;2. Write a program that stores the given two 32-bit numbers into the

;current data segment at offset: 0x1000 and 0x1008, respectively.

;The program then calculates the sum of these numbers and stores it at

;the offset: 0x1010.

;Numbers: 0x1F540398, 0xC0A1F02E.

.data

num1 dw 0x0398 ;lower word

num2 dw 0x1f54 ;upper word

num3 dw 0xf02e ;lower word

num4 dw 0xc0a1 ;upper word

.code

main proc

mov ax, @data

mov ds, ax

mov bx, 0x1000 ;offset

;store in memory

mov si, 0

mov ax, num2

mov [bx + si], ax

add si, 2

mov ax, num1

mov [bx + si], ax

add si, 6

mov ax, num4

mov [bx + si], ax

add si, 2

mov ax, num3

mov [bx + si], ax

;load lower words from memory and sum

mov si, 2

mov ax, [bx + si]

add si, 8

mov dx, [bx + si]

add ax, dx

mov [bx + 0xc], ax

;load lower words from memory and sum

mov si, 0

mov ax, [bx + si]

add si, 8

mov dx, [bx + si]

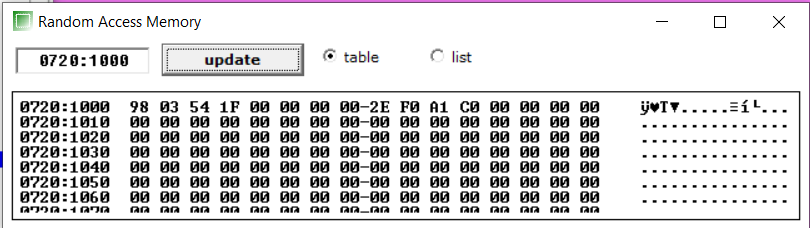
adc ax, dx

mov [bx + 0xa], ax

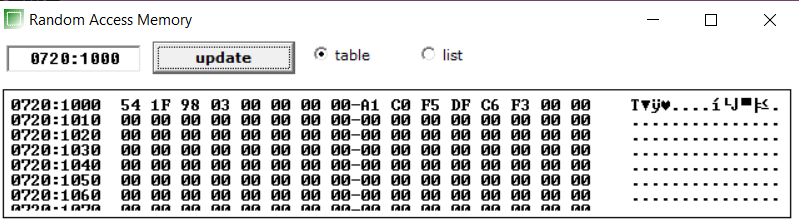
main endp

.exit

Memory:



Output stored in memory:



1. Write code that set both PF and ZF.

Code:

.model small

.stack 100h

;3. Write code that set both PF and ZF.

.data

.code

main proc

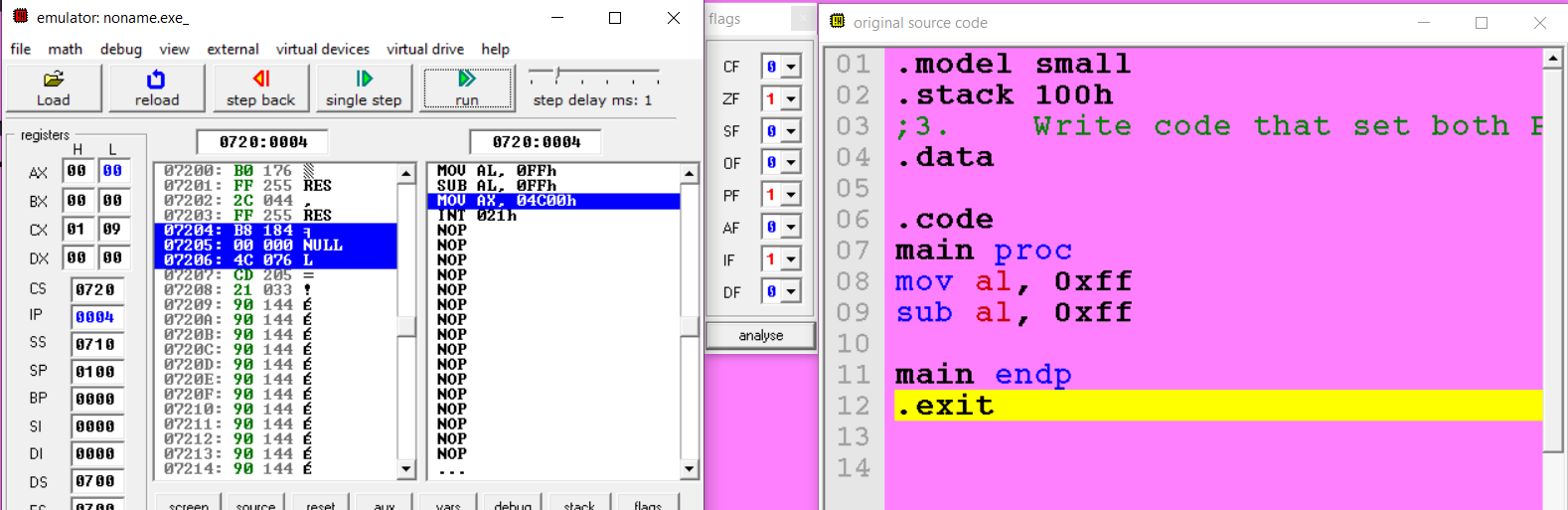
mov al, 0xff

sub al, 0xff

main endp

.exit

Flags:



1. Write a program that sub two 16-bit numbers stored in memory at offset x2000h and checks if ZF is set. If ZF = 1, store 1 at memory location 4000h; otherwise, store at 0x1000h.(use jmp)

Code:

.model small

.stack 100h

;4. Write program that sub two 16-bit numbers stored in memory at offset x2000h

;and checks if ZF is set. If ZF = 1, store 1 at memory location 4000h;

;otherwise, store at 0x1000h.(use jmp)

.data

num1 dw 0x1234

num2 dw 0x1234

.code

main proc

mov ax, @data

mov ds, ax

mov bx, 0x2000 ;offset

;store in memory

mov ax, num1

mov [bx], ax

mov ax, num2

mov [bx + 2], ax

;load from memory and subtract

mov ax, [bx]

sub ax, [bx + 2]

jz mov\_to\_0x4000

mov [bx - 0x1000], ax

jmp end

mov\_to\_0x4000:

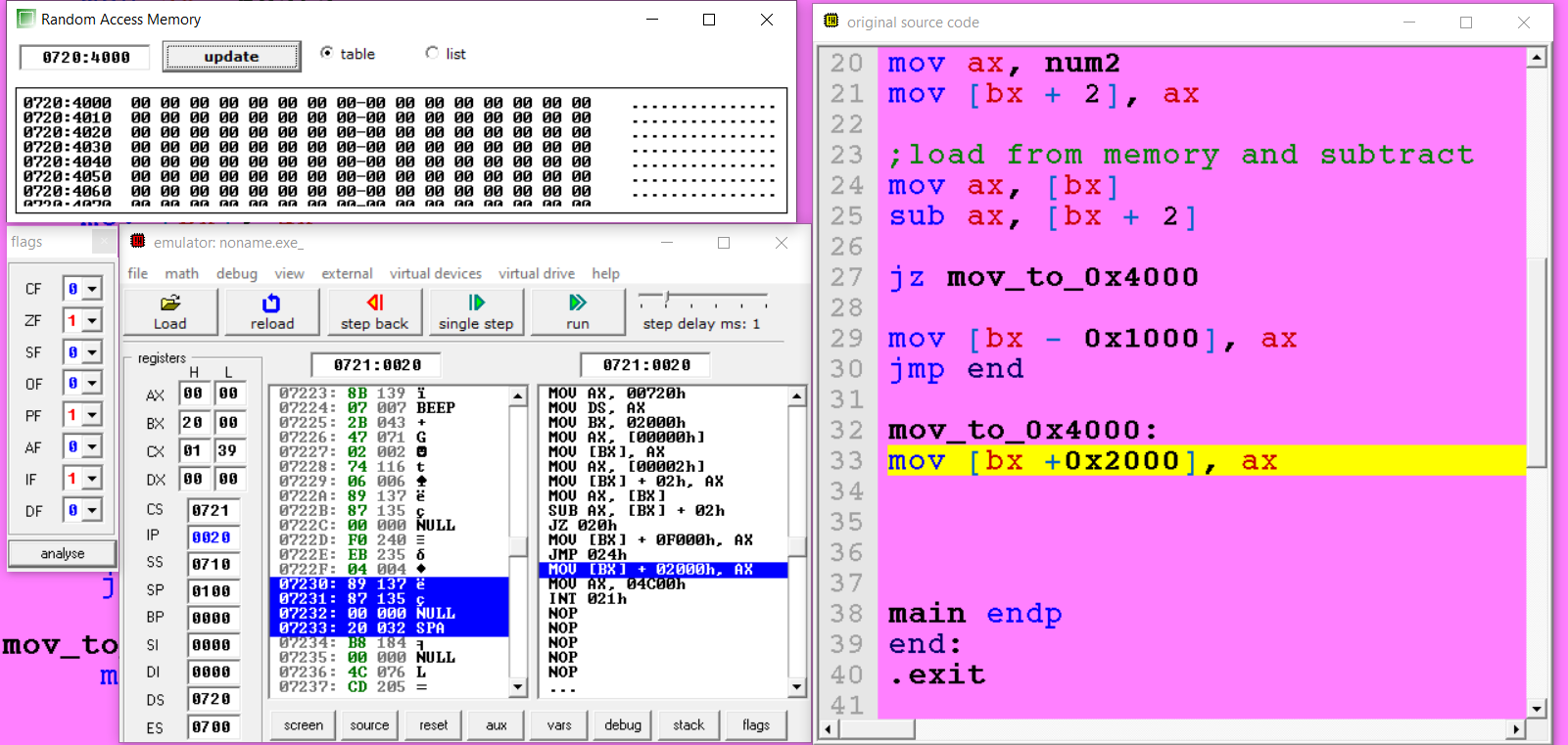
mov [bx + 0x2000], ax

main endp

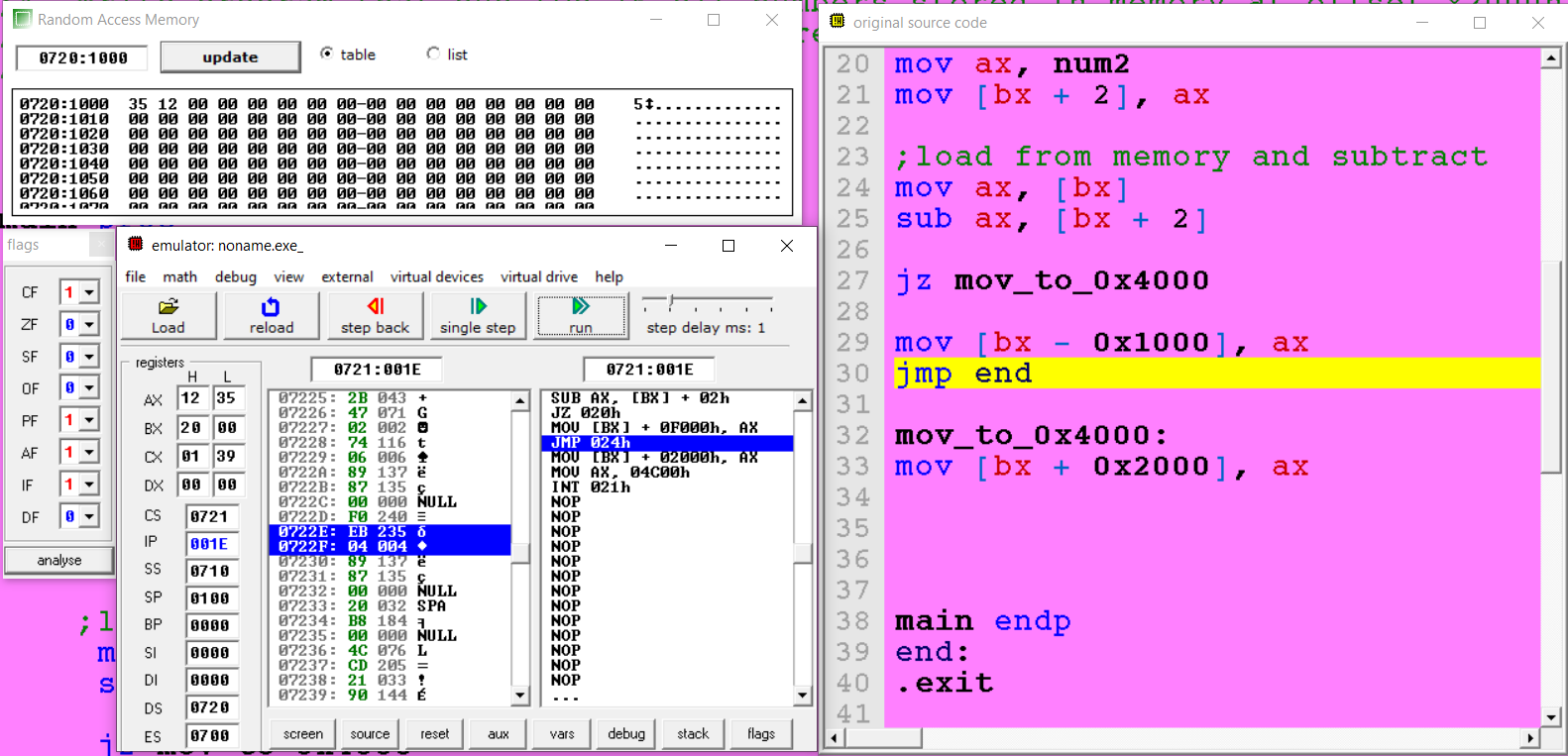
end:

.exit

ZF is set:



ZF is not set:



5. Write a program to implement the following equation.

X = ~0xFF12 ^ {0xABFF & (0x2113 | 0x2340)}

~ Invert all bits

^ Bitwise XOR

& Bitwise AND

| Bitwise OR

Code:

.model small

.stack 100h

.data

num1 dw 0xFF12

num2 dw 0xABFF

num3 dw 0x2113

num4 dw 0x2340

x dw 0x0000

.code

main proc

mov ax, @data

mov ds, ax

mov ax, num3

mov dx, num4

or ax, dx

mov dx, num2

and ax, dx

mov dx, num1

xor ax, dx

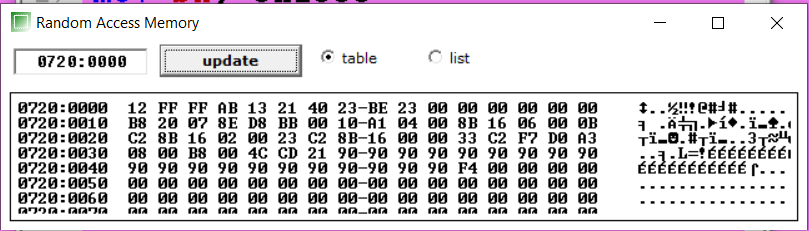
not ax

mov x, ax

main endp

.exit

Storred in memory:



6. Write a program that stores a 32-bit number and a 16-bit number into the current data segment at offsets 0x1000 and 0x1008, respectively. The program then subs these two numbers. 0xC0A1F02E, 0x45FA

Code:

.model small

.stack 100h

;Write a program that stores a 32-bit number and a 16-bit number into the

;current data segment at offsets 0x1000 and 0x1008, respectively.

;The program then subs these two numbers. 0xC0A1F02E, 0x45FA

.data

num1 dw 0xF02E ;lower word

num2 dw 0xC0A1 ;upper word

num3 dw 0x45FA ;second number

lresult dw 0x0000

uresult dw 0x0000

.code

main proc

mov ax, @data

mov ds, ax

mov bx, 0x1000 ;offset

;store in memory

mov si, 0

mov ax, num2

mov [bx + si], ax

add si, 2

mov ax, num1

mov [bx + si], ax

add si, 6

mov ax, num3

mov [bx + si], ax

;load from memory and subtract

mov si, 2

mov ax, [bx + si]

add si, 6

mov dx, [bx + si]

sub ax, dx

;store in result variable

mov dx, [bx]

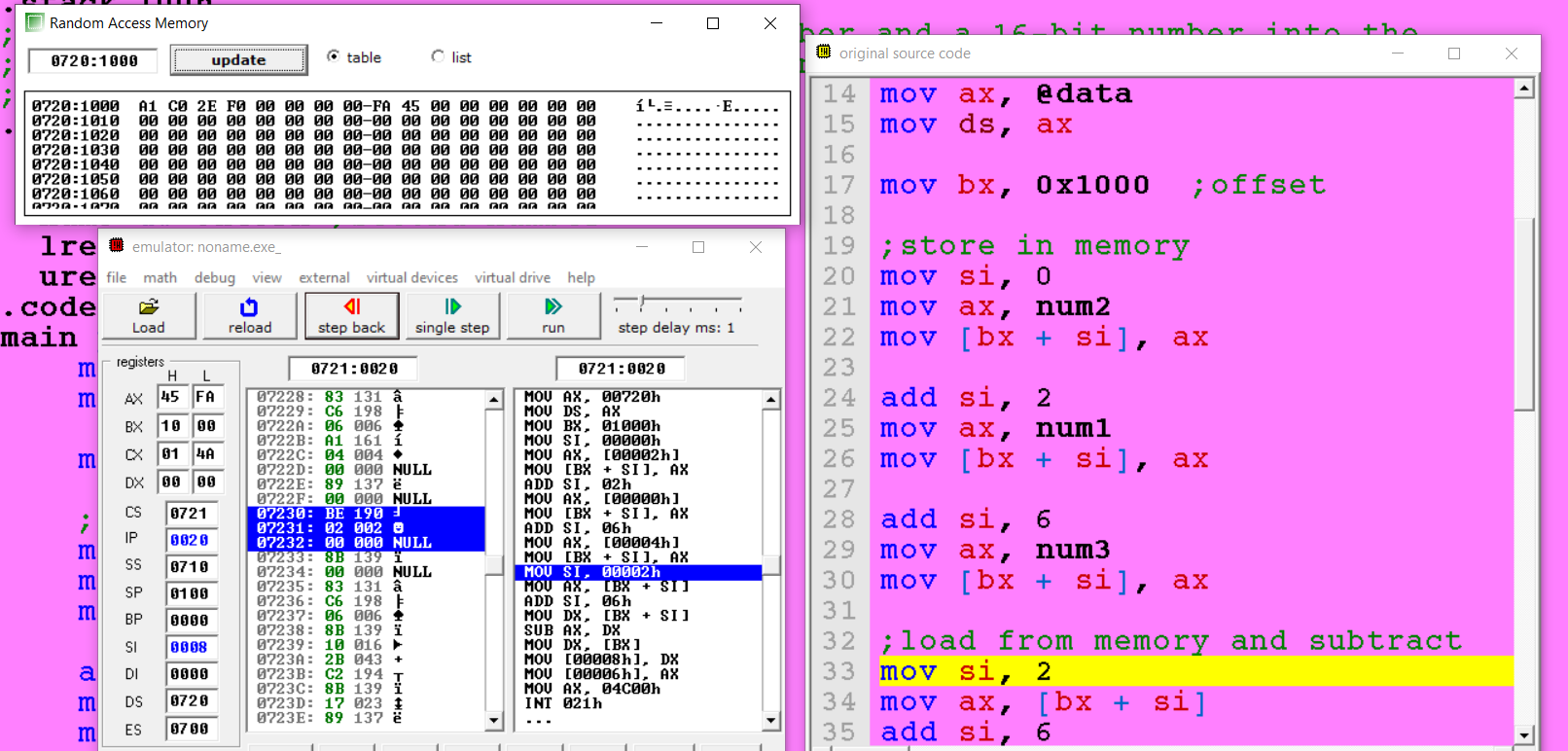
mov uresult, dx ;stored at 0720:0000 memory(global variables)

mov lresult, ax ;stored at 0720:0000 memory(global variables)

main endp

.exit

Stored in memory:



Stored in result variable in data segment:

A screenshot of a computer program

AI-generated content may be incorrect.

7. Write an assembly program to store **5 numbers** in an array and then add them.

Code:

.model small

.stack 100h

;7. Write an assembly program to store 5 numbers in an array and then add them.

.data

array dw 0xaaaa, 0xbbbb, 0xcccc, 0xdddd, 0xeeee

sum dw 0x0000

.code

main proc

mov ax, @data

mov ds, ax

mov cx, 0x0005

mov si, offset array

mov ax, 0

loop:

add ax, [si]

add si, 2

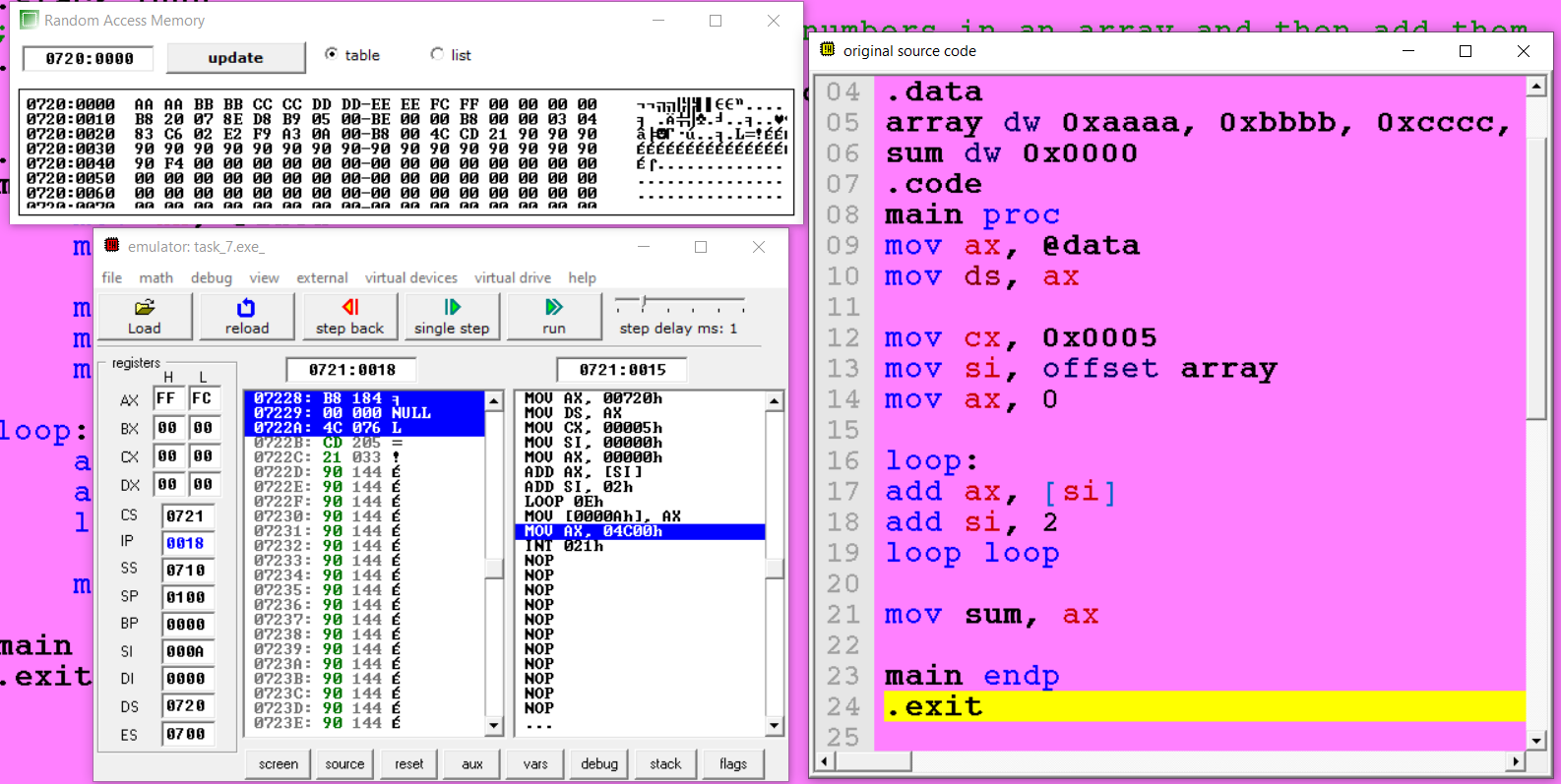
loop loop

mov sum, ax

main endp

.exit

Saved in memory:



8. Take a variable in .data section, and calculate its table. (use loops).

Code:

.model small

.stack 100h

;8. Take a variable in .data section, and calculate its table. (use loops).

.data

num dw 0x0005

.code

main proc

mov ax, @data

mov ds, ax

mov ax, num

mov cx, 0x0009

;loop runs 9 times

loop:

add ax, 0x0005

loop loop

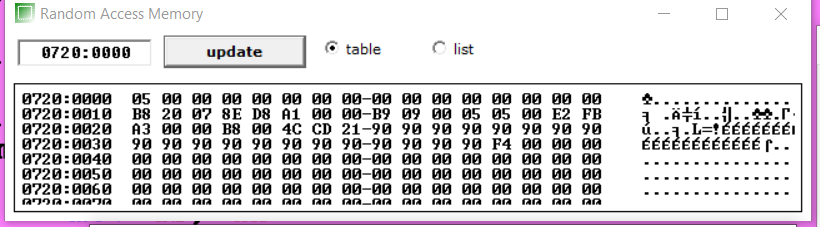
;save in memory variable

mov num, ax

main endp

.exit

Number 0x0005:



Last value after loop running to print table:

